REMARKS

Claims 1-41 were pending as of the action mailed August 14, 2003. Claims 10, 12, 24-27, 30, 34, 37, and 41 are canceled. New dependent claims 42-57 are submitted. No new matter has been added. Reconsideration of the action in light of the foregoing amendments and the following remarks is respectfully requested.

Claim 11 was rejected under 35 U.S.C. § 112, second paragraph, as being indefinite. The claim has been amended to address the Examiner's rejection without changing the scope of the claim.

Claims 1, 2, 5-7, 9, 10, 28-30, and 35-37 were rejected as unpatentable over U.S. Patent No. 5,404,428 ("Wu") in view of U.S. Patent No. 5,929,864 ("Picott").

Independent claims 1, 28, and 35 have corresponding limitations. The following remarks apply to each of them and all of their dependent claims. Claim 1 recites a method in which "when the value of object B changes, . . . severing dependencies among the dependents of object B and all of their further dependents." In the action, the Examiner does not mention this feature (see pages 2 and 3 of the action); nor is this feature found in Wu or Picott.

In discussing claim 5, which depends from claim 1, the Examiner states that "Wu teaches breaking any dependency relationships an object may have had when the object is marked as being dirty", citing column 8, lines 11-43 of Wu. A careful review of this cited passage shows nothing that indicates that the dependency relationship between derived data and context is ever changed; and more particularly, nothing indicates that the dependency relationship is ever severed. Further as to claim 5 the Examiner states that "identifying the objects on which the recomputed value depends and as dependent only on the identified object when the value of the object is re-computed" is also found in Wu, citing column 9, lines 8-41. What the claim says is that the method includes "identifying the objects on which the recomputed value is actually dependent and identifying the recomputed object as dependent only on the identified objects". A careful reading of the passage cited by the Examiner shows that this feature is not found in Wu. In fact, the cited passage describes a system in which a static dependency graph exists in the form of the acyclic directed graph described in Wu, column 9, lines 42-61. Thus, unlike the claimed invention, in which dependencies are changed dynamically and depend on the state of

the application, the dependencies in Wu are static. For at least the foregoing reasons, claims 1, 28 and 35, and their dependent claims, are allowable.

Claims 3 and 4 were rejected as unpatentable over Wu in view of Picott, and further in view of U.S. Patent No. 5,251,290 ("Pabon").

Claims 3 and 4 depend from claim 1 and are allowable for at least the reasons set forth above.

Independent claims 7, 29, and 36 are subject to a common rejection. The Examiner finds the limitation "identifying the objects upon which a given object depends as those objects into which the given object passed itself as a requestor during execution of a compute method of the given object" (claim 7) in Wu, column 9, lines 1-7. The cited passage of Wu actually reads as follows:

> "Each derived item declined in the system in implemented embodiments contains references, in the form of direct function calls for evaluating other items which have validity flags. In this way, when a view model attribute is modified by the application program, derived items dependent upon that attribute change, are invalidated during the change."

Nothing in the cited passage or anywhere else in Wu teaches an object that passes itself as a requestor during the execution of a compute method. For at least this reason, claims 7, 29, and 36, and their dependent claims, are allowable.

Claim 8 was rejected as unpatentable over Wu in view of Picott, and further in view of U.S. Patent No. 5,689,711 ("Bardasz"). Claim 8 depends from claim 7 and is allowable for at least the reasons set forth above.

Independent claims 10, 30, and 37 have been cancelled.

Claims 11-27, 31-34, and 38-41 were rejected as unpatentable over U.S. Patent No. 6,505,228 ("Schoening") in view of U.S. Patent No. 6,128,771 ("Tock").

Claim 11 has been amended to incorporate the limitation of claim 12. Claims 11 and 12 as filed were rejected by reference to the rejection of claim 15. In the rejection of claim 15, the Examiner finds the feature of "detaching the object" in Schoening, citing 'markDirty method 708 Col. 26 Ln. 60-67, Col. 27 Ln. 1-24, "...markDirty() function..." Col. 52 Ln. 35-54).' The

applicant respectfully submits that the feature in question is not in fact described or suggested by the cited passages, which read as follows:

> "Preferably, in the ANI 50, one or more Persistent Objects are integrated with the Service Module Functions 76a-76n. In particular, the writer of a Service Module Function 76a-76n configures the Service Module Function object or helper object to declare its metadata as a POMetaData object 715. When each Service Module Function 76a-76n sets values of persistent data objects, it invokes the markDirty method 708 to indicate that the values of the objects have changed. Each Persistent Object 80 has a metaData () method 714 that returns the POMetaData object 715, and implements a writeObject method 712 and readObject method 712 to write and read the persistent data.
>
> In the preferred embodiment, the Dirty method 706 returns a Boolean value. The Boolean value is TRUE when the persistent data in the object has changed since the last load or store operation, and FALSE when persistent data in the object has not changed since the last load or store.
>
> The Mark Dirty method 708 sets an object's "dirty" state to TRUE, indicating that persistent data in this class has changed. Preferably in each persistent object, a Dirty instance variable indicates the current state of the object's persistent data objects. When object values are initially loaded from the database, each associated persistent object is marked "clean" by setting Dirty FALSE. Similarly, when the persistent data objects are stored to the database, the persistent object is also marked "clean".
>
> The "dirty" state determines whether to actually store the contents of an object to the database 60 when the Write Object method 712 is invoked for that object. In the preferred embodiment, when it is time to update secondary storage all objects which were marked dirty are written to the database 60."

As is clear from a careful reading of the foregoing paragraphs, nowhere in the cited passages does Schoening teach or suggest the limitation of "severing dependencies from the changed object and all of its direct and indirect dependent objects", as required by amended claim 11. For at least the foregoing reason, claim 11 and its dependent claims are allowable.

Claims 31 and 38 have been amended and recite limitations corresponding to those of claim 11. Thus, for all the reasons applicable to claim 11, claims 31 and 38 and their dependent claims are allowable.

Claim 15 recites a method that includes "detaching the dependent object from the dependency graph". As has just been explained with reference to claim 11, this feature is not found in Schoening.

In addition, claim 15 recites "rejoining recomputed objects with the dependency graph, whereby leaf objects are rejoined with the dependency graph." As is clear from a careful reading of the cited passages from Schoening quoted above, Schoening nowhere teaches or suggests this feature. For at least the foregoing reasons, claim 15 and its dependent claims are allowable.

Claims 32 and 39 recite limitations corresponding to those of claim 15. Thus, for all the reasons applicable to claim 15, claims 32 and 39 and their dependent claims are allowable.

Claim 19 recites a method that includes "calculating the dependency among objects in the set dynamically at the time objects calculate their values." Claims 33 and 40 have corresponding limitations. Claims 19 and 40 were rejected by reference to the rejection of claim 33. In the rejection of claim 33, the Examiner states that "means for calculating the dependency among objects in the set dynamically at the time objects calculate their values" is found in column 51, lines 56-67 of Schoening. The applicant respectfully disagrees. The passage in which the Examiner relies reads in its entirety:

> "5. Transactions
>
> The preferred embodiment includes a transaction processing mechanism that manages changes to data modifications. Transactions provide several following services. For example, transactions collect objects that are modified during the transaction's lifetime within the process being managed. Transactions synchronize access to objects being modified between transactions. Transactions provide for database update of modified objects. Transactions provide for database update of modified objects. Transactions provide for event generation describing changes made during a transaction."

Nothing in the cited passage teaches or suggests the feature of "calculating the dependency . . . dynamically . . ." as required by claims 19, 33, and 40. For at least this reason, claims 19, 33, and 40 and their dependent claims are allowable.

Enclosed is a check in the amount of $936. for excess claim fees and a two month extension of time. Please apply any other charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: 08 Jan 04

Hans R. Troesch
Reg. No. 36,950

Fish & Richardson P.C.
500 Arguello Street, Suite 500
Redwood City, California 94063
Telephone: (650) 839-5070
Facsimile: (650) 839-5071

50171569.doc